Constructing a data dictionary and appending it to your data in R

Tomas Ko

11 July, 2022

Overview

A data dictionary is an important tool in data management. A data dictionary is a supplementary document that details the information about variables, data collection, and other important features of a data set, i.e. metadata, data that describes other data. This metadata is crucial and helps others find, access, understand, and reuse your data. Without proper documentation, the data you store in online repositories may be rendered unfindable and unusable by others and indexing search engines. The *dataMeta* R package is designed to create a data dictionary and append it to the original dataset's attributes list along with other information generally provided as metadata. In this lesson, we will use the *dataMeta* R package to construct a data dictionary for a subset of the NHS England accident and emergency (A&E) attendances and admissions (`ae_attendances`) data from the *NHSRdatasets* package; we will then map the data and then save it as an R dataset (.rds) in your 'RawData' folder.

Load packages and read in the data

Let's load the packages and data needed for this document. The *dataMeta* packages is not installed on Noteable. Therefore, you will need to `install.packages('dataMeta')` every time you want to knit this document in RStudio in Notable. We will use the `read_csv()` function from the *readr* package from *tidyverse* to read in the data. *tidyverse* is a collection of essential R packages for data science. The *readr* package provides a fast and friendly way to read rectangular data from delimited files, such as comma-separated values (CSV) and tab-separated values (TSV). The * readr * package is loaded by the *tidyverse* package as one of its core components. We will use the *here* package to build a path relative to the top-level directory to read the raw ae_attendances data from our 'RawData' folder. The *here* package enables easy file referencing in project-oriented workflows. In contrast to using `setwd()` function, which is fragile and dependent on the way you organise your files, here uses the top-level directory of a project to easily build paths to files. The *lubridate* provides tools that make it easier to manipulate dates in R.

```
library(dataMeta)
library (tidyverse)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.6
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(here)
```

```
## here() starts at /home/jovyan/B202280/B202280_assessment
```

Data

The data you will be managing on the course are from the NHSRdatasets package. This package has been created to support skills development in the NHS-R community and contains several free datasets. The dataset set I have chosen to manage from the NHSRdatasets package is the NHS England accident and emergency (A&E) attendances and admissions (`ae_attendances`) data. The `ae_attendances` data includes reported attendances, four-hour breaches and admissions for all A&E departments in England for 2016/17 through 2018/19 (Apr-Mar). We previously selected a subset of the variables needed for my data capture tool, including period, attendances and breaches, and subsetted the data into test and training data. However, for this lesson, we will use the data collected from the full `ae_attendances` dataset to demonstrate how to use the *dataMeta* to construct a data dictionary and append it to your collected data. The Jupyter Notebook "./ipynbScripts/CollectingDataUsingInteractiveJupyterWidgets.ipynb" was used to to collect the data.

Note, you only need to construct and append of data dictionary for the subset of the variables required for your data capture tool. We are using the full data set here, as you will be using the *dataMeta* R package construct and append a data dictionary for different variables from your `ae_attendances` data subset collected by your data capture tool.

Let us use the `read_csv()` function from the *readr* package to read your collected data from the Raw data folder.

```
CollectedData=read_csv(here("RawData", "CollectedDataFinal.csv"))
```

```
## Rows: 1 Columns: 4

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): index, attendances
## lgl  (1): consent
## date (1): period

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Let's view the CollectedData ae_attendances data

The `glimpse()` function is from *tibble* package and is used to view the columns/variables in a data frame. It also shows data type and some of the data in the data frame in each row. The *tibble* package provides utilities for handling tibbles. The *tibble* package is loaded by the *tidyverse* package, as one of its core components.

```
glimpse(CollectedData)
```

```
## Rows: 1
## Columns: 4
## $ index       <dbl> 1155
## $ period      <date> 2016-12-01
## $ attendances <dbl> 200
## $ consent     <lgl> TRUE
```

Here is the output of the 'glimpse()' function. It starts off with the number of rows and columns and each column in separate rows.

The CollectedData dataset contains:

- **index:** the index column that allows us to link the data collected to the original ae_attendances data in the 'RawData' folder.
- **period:** the month that this activity relates to, stored as a date (1st of each month).
- **attendances:** the number of attendances for this department type at this organisation for this month.
- **consent:** the consent from the end-user to process and share the data collected with the data capture tool.

Build a data dictionary for the data collected by the data capture tool

Build a linker data frame

We first need to build a linker data frame. To do this, we need to create two string vectors representing the different variable descriptions and the different variable types.

Variable descriptions

We need to create a string vector representing the different variable descriptions.

```
variable_description <- c("The index column that allows us to link the data collected to the original ae
"The month that this activity relates to, stored as a date (1st of each month).",

"The number of attendances for this department type at this organisation for this month.",

"The consent from the end-user to process and share the data collected with the data capture tool.")
print(variable_description)
```

```
## [1] "The index column that allows us to link the data collected to the original ae_attendances data i
## [2] "The month that this activity relates to, stored as a date (1st of each month)."
## [3] "The number of attendances for this department type at this organisation for this month."
## [4] "The consent from the end-user to process and share the data collected with the data capture tool
```

Variable types

We need to create a string vector representing the different variable types. It is a vector of integers with values 0 or 1. We need to use 0 for a variable with quantitative values (measured values) variables and 1 for fixed values (allowable values or codes) variables. Let us use The `glimpse()` function from *tibble* package to view the variable types in the CollectedData data frame.

```
#Data meta does not accept date variable so we are changing date variable to a character format.
CollectedData$period<-as.character(CollectedData$period)
glimpse(CollectedData)
```

```
## Rows: 1
## Columns: 4
## $ index       <dbl> 1155
## $ period      <chr> "2016-12-01"
## $ attendances <dbl> 200
## $ consent     <lgl> TRUE
```

We have five quantitative values (measured values) variables and four fixed values (allowable values or codes) variables.

```
variable_type <- c(0, 1, 0, 1)
print(variable_type)
```

```
## [1] 0 1 0 1
```

Now let us use the `build_linker()` function from the *dataMeta* package to constructs an intermediary (linker) data frame between the CollectedData and the data dictionary. For this function to run, it requires the CollectedData data frame and variable_description and variable_type string vectors as inputs.

```
linker<-build_linker(CollectedData, variable_description, variable_type)
print(linker)
```

```
##      var_name
## 1       index
## 2      period
## 3 attendances
## 4     consent
##
## 1 The index column that allows us to link the data collected to the original ae_attendances data in t
## 2                                                 The month that this activity relates to, stored as a date
## 3                                        The number of attendances for this department type at this organisa
## 4                              The consent from the end-user to process and share the data collected with th
##   var_type
## 1        0
## 2        1
## 3        0
## 4        1
```

Data dictionary

We are now going to use the `build_dict()` function from the *dataMeta* to constructs a data dictionary for a CollectedData data frame with the aid of the linker data frame between. For this function to run, it requires the CollectedData and linker data frames and variable_description as inputs.

```
dictionary <- build_dict(my.data = CollectedData, linker = linker)
```

```
## Enter description for variable 'attendances' and option '200 to 200':
## Enter description for variable 'consent' and option 'TRUE':
## Enter description for variable 'index' and option '1155 to 1155':
## Enter description for variable 'period' and option '2016-12-01':
```

```
glimpse(dictionary)
```

```
## Rows: 4
## Columns: 4
## $ `variable name`        <chr> "attendances", "consent", "index", "period"
## $ `variable description` <chr> "The number of attendances for this department ~
## $ `variable options`     <chr> "200 to 200", "TRUE", "1155 to 1155", "2016-12-~
## $ notes                  <chr> "", "", "", ""
```

Let's save the data dictionary for CollectedData to the 'RawData' folder

Of note, when naming folders and files, you must do so in a consistent, logical and predictable way means that information may be located, identified and retrieved by your and your colleagues as quickly and efficiently as possible. With this in mind, let's name this file "CollectedData_DataDictionary" and write it to the raw data folder.

```
glimpse(dictionary)
```

```
## Rows: 4
## Columns: 4
## $ `variable name`        <chr> "attendances", "consent", "index", "period"
## $ `variable description` <chr> "The number of attendances for this department ~
## $ `variable options`     <chr> "200 to 200", "TRUE", "1155 to 1155", "2016-12--~
## $ notes                  <chr> "", "", "", ""
```

```
write_csv(dictionary, here("RawData", "CollectedData_DataDictionary.csv"))
```

Append data dictionary to the CollectedData

We will now incorporate attributes as metadata to the CollectedData as metadata using the 'incorporate_attr()' function from the *dataMeta* package. For this function to run, it requires the CollectedData and dictionary and main_string main_string as inputs. main_string is a character string describing the CollectedData data frame.

```

main_string <- "This data describes the NHS England accident and emergency (A&E) attendances and breache
main_string
```

Create main_string for attributes

```
## [1] "This data describes the NHS England accident and emergency (A&E) attendances and breaches of fou
```

Incorporate attributes as metada We are using the 'incorporate_attr()' function to return an R dataset containing metadata stored in its attributes. The attributes we are going to add include: * a data dictionary * number of columns * number of rows * the name of the author who created the dictionary and added it, * the time when it was last edited * a brief description of the original dataset.

```
complete_CollectedData <- incorporate_attr(my.data = CollectedData, data.dictionary = dictionary,
main_string = main_string)
#Add the authors name.
attributes(complete_CollectedData)$author[1]<-"Tomas Ko"
complete_CollectedData
```

```
## # A tibble: 1 x 4
##   index period     attendances consent
## * <dbl> <chr>            <dbl> <lgl>
## 1  1155 2016-12-01         200 TRUE
```

```
attributes(complete_CollectedData)
```

```
## $row.names
## [1] 1
##
## $names
## [1] "index"       "period"      "attendances" "consent"
##
## $spec
## cols(
##   index = col_double(),
##   period = col_date(format = ""),
##   attendances = col_double(),
##   consent = col_logical()
## )
##
## $problems
## <pointer: 0x55e539e891c0>
##
## $class
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## $main
## [1] "This data describes the NHS England accident and emergency (A&E) attendances and breaches of fou
##
## $dictionary
##   variable name
## 1   attendances
## 2       consent
## 3         index
## 4        period
##
## 1                                  The number of attendances for this department type at this organisa
## 2                        The consent from the end-user to process and share the data collected with th
## 3 The index column that allows us to link the data collected to the original ae_attendances data in t
## 4                                           The month that this activity relates to, stored as a date
##   variable options notes
## 1       200 to 200
## 2             TRUE
## 3     1155 to 1155
## 4       2016-12-01
##
## $last_edit_date
## [1] "2022-07-11 10:02:46 UTC"
##
## $author
## [1] "Tomas Ko"
```

Save the CollectedData with attributes We are using the 'save_it()' function to save the CollectedData with attributes stored as metadata as an R dataset (.rds) into the 'current working directory'RawData' folder. This is the final function used in this package. For the function to run, the complete_CollectedData, and the name of the file as a text string to name the file are required as inputs.

```
save_it(complete_CollectedData, here("RawData", "complete_CollectedData"))
```

If you would like to load this data later, here is the code to do so

```
complete_CollectedData<-readRDS(here("RawData", "complete_CollectedData.rds"))
```

Well done! You now have now saved your CollectedData_DataDictionary and enriched Collected-Data_DataDictionary.rds file to your 'Rawdata' folder. Happy coding!